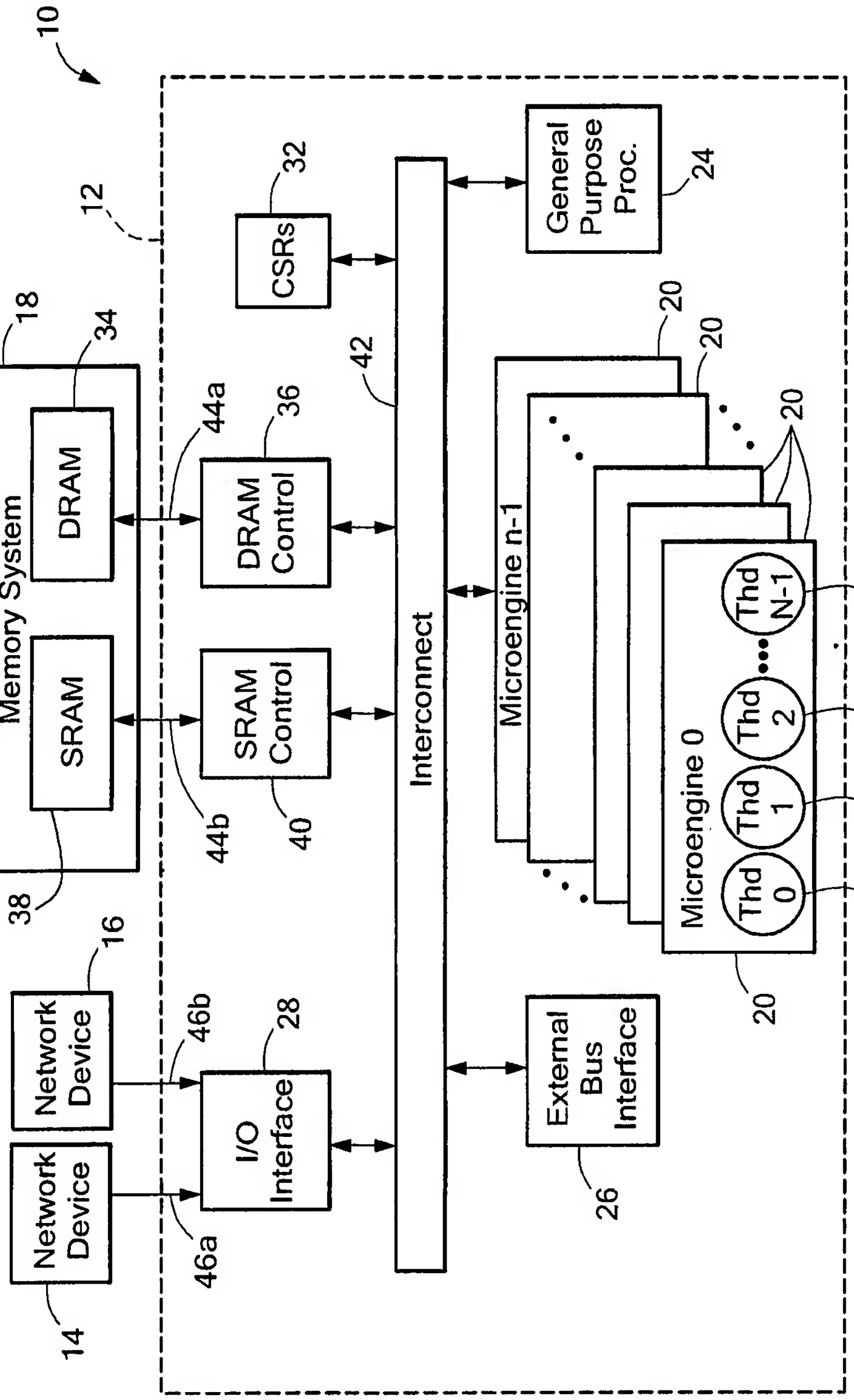

FIG. 1

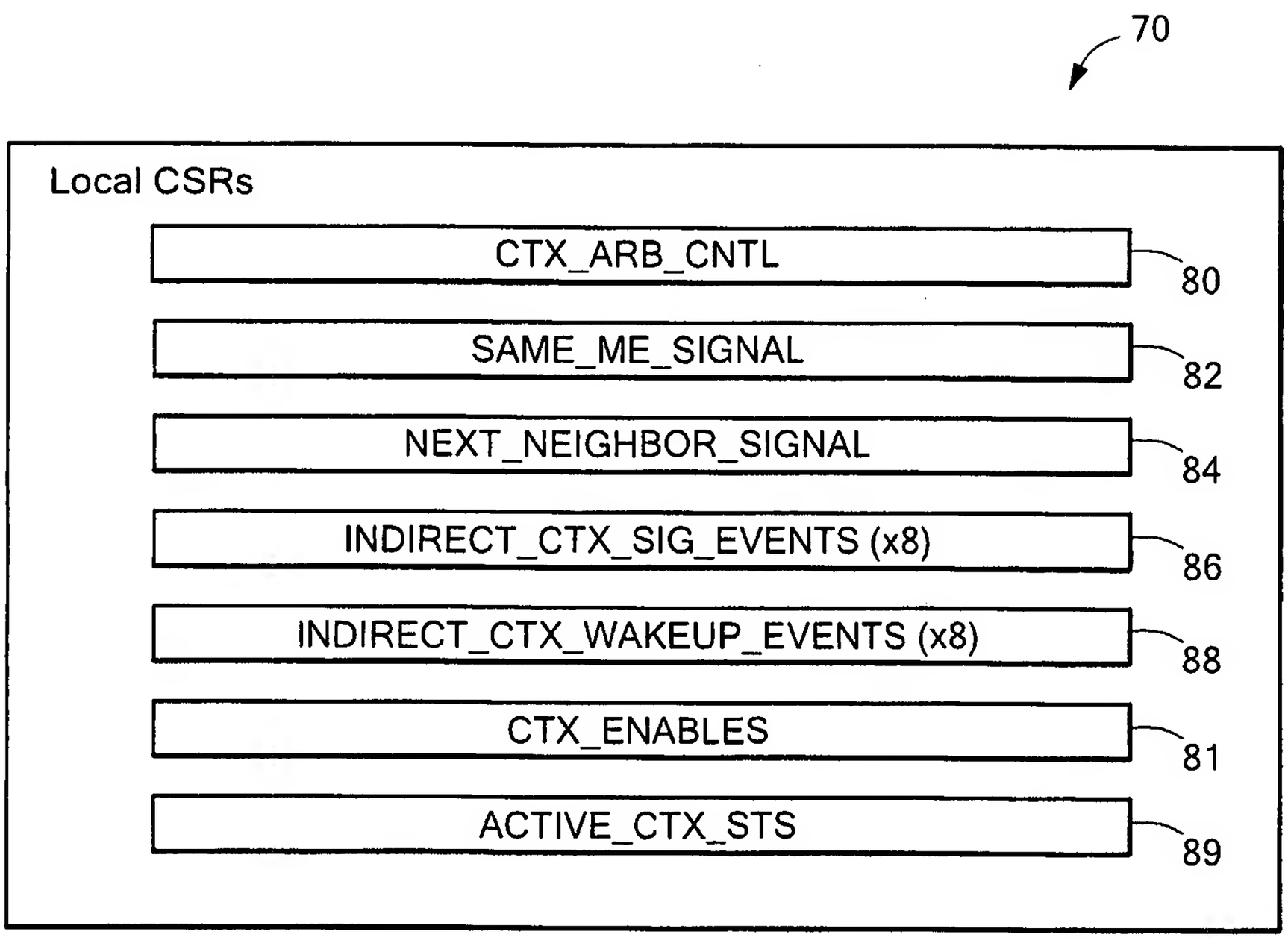
70
Local CSRs
CTX_ARB_CNTL
80
SAME_ME_SIGNAL
82
NEXT_NEIGHBOR_SIGNAL
84
INDIRECT_CTX_SIG_EVENTS (x8)
86
INDIRECT_CTX_WAKEUP_EVENTS (x8)
88
CTX_ENABLES
81
ACTIVE_CTX_STS
89

FIG. 3

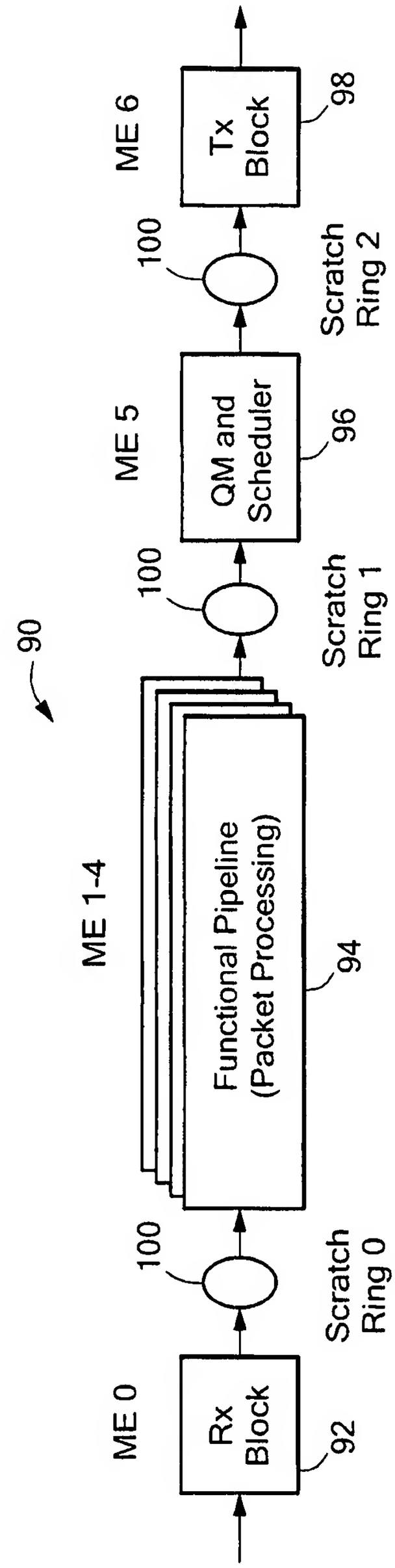
90
ME 1-4
Functional Pipeline (Packet Processing)
94
100
Scratch Ring 0
ME 0
Rx Block
92
100
Scratch Ring 1
ME 5
QM and Scheduler
96
100
Scratch Ring 2
ME 6
Tx Block
98

FIG. 4

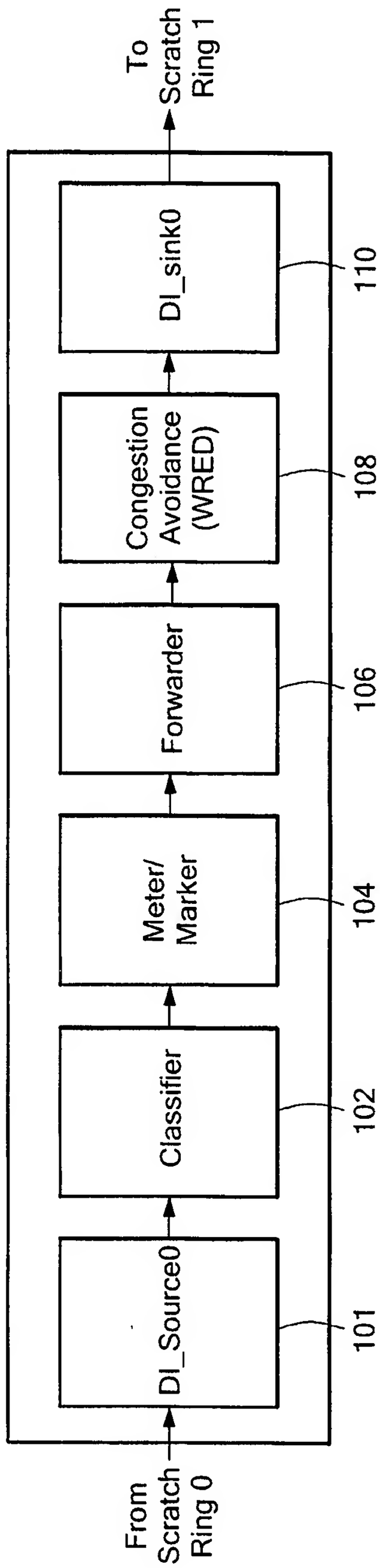
94
From Scratch Ring 0
DI_Source0
101
Classifier
102
Meter/ Marker
104
Forwarder
106
Congestion Avoidance (WRED)
108
DI_sink0
110
To Scratch Ring 1

FIG. 5

120

Register Name	Latency in terms of instructions		Usage Latency Comments
	Write	Read	Usage
SAME_ME_SIGNAL 82	3 122	2 124	8 126

FIG. 6

```
wred 0                                                                    130
{
        if(ctx0=0)                                                        ┐
        {                                                                 │
        //Wait for signal from previous ME and thread 7                   │
                                                                          │
        wait_for_all (&next_thread_signal, &wred_next_me_sig);            │
             cam_clear 0;                                                 │
        }                                                                 │
        else                                                              │
        {                                                                 │
             wait_for_all (&next_thread_signal);                          │
        }                                                                 ┘ 132
        .....
        .....
        //WRED packet processing
        .....                                                    134
        signal_next_thread 0 //Instruction 1                     ◄

        .....
        //There is a minimum of 3 cycles delay between instruction 1 and instruction 2
        //to allow the signal to propagate and to ensure thread execution sequence.
        .....                                                    136
        //Wait for previous thread signal                        ◄
        wait_for_sig (&sig);   //Instruction 2
        .....
        if(ctx 0=7)                                                       ┐
        {                                                                 │
             //Signal next ME                                             │
                                                                          │
             cap_fast_write (wred_me_sig_csr, csr_interthread_sig);       │
        }                                                                 ┘ 138
        else
        {
             //The thread give up the context voluntarily at this point to ensure that
             //thread 7 gets control as early as possible. If no context swap occurs
             //here the thread would continue to execute non-critical section code or
             //       next microblock, thereby delaying thread 7 getting the control.
             ctx_arb(voluntary);                                 ◄
        }                                                        140

        //Critical section processing ends

        //Non-critical section code or code for next microblock begins
```

FIG. 7

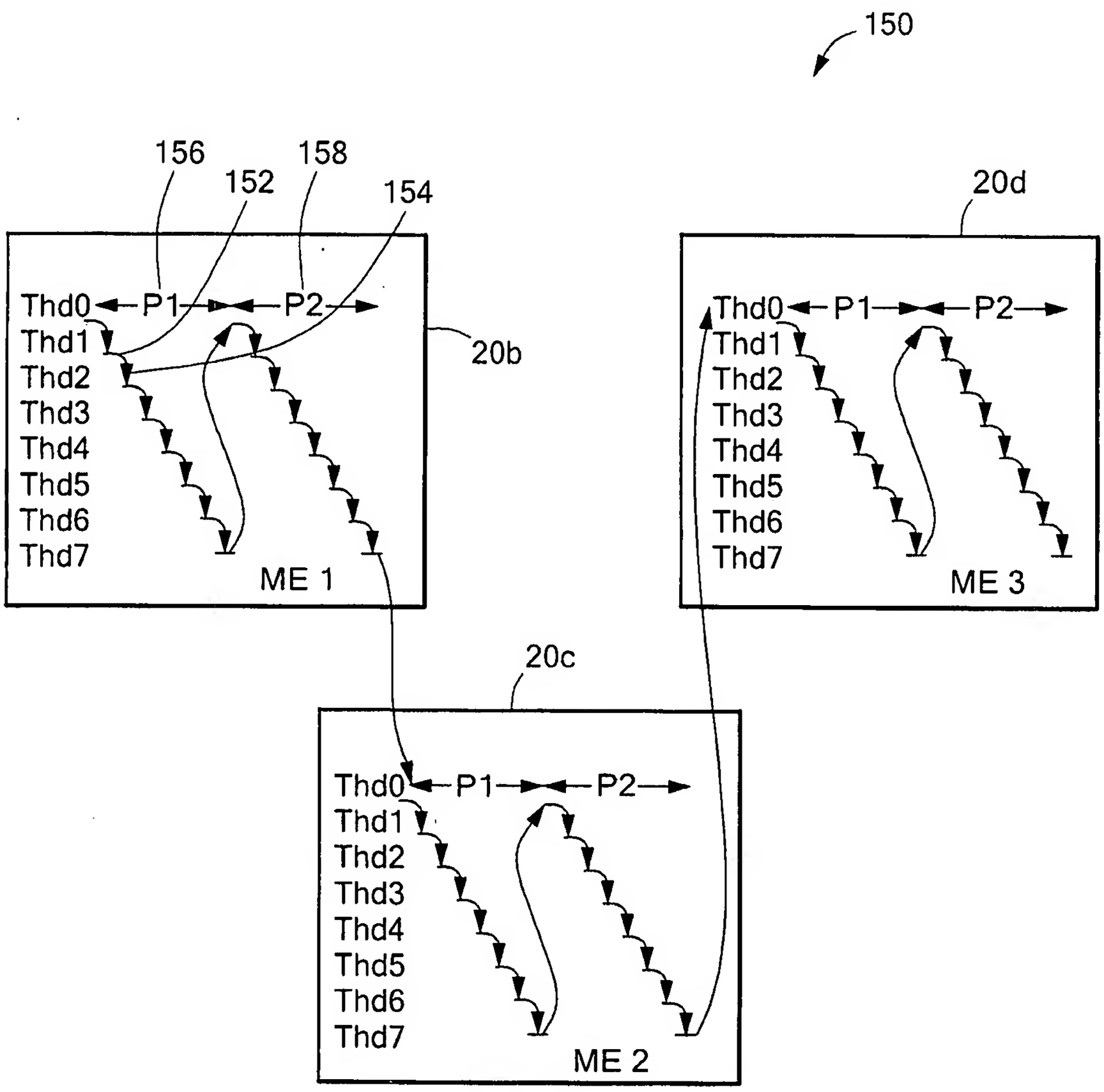
150
156
158
152
154
20d
Thd0
P1
P2
Thd1
Thd2
Thd3
Thd4
Thd5
Thd6
Thd7
20b
ME 1
ME 3
20c
ME 2

FIG. 8

14/14

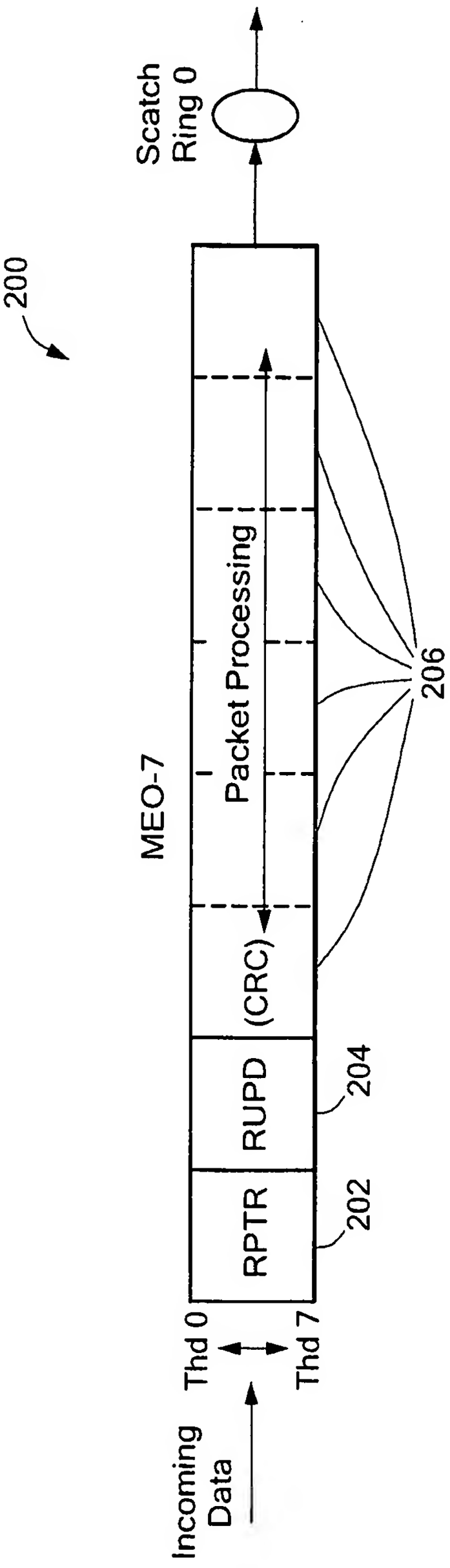
Scatch Ring 0
200
MEO-7
Packet Processing
206
(CRC)
RUPD
204
RPTR
202
Thd 0
Thd 7
Incoming Data

FIG. 13